Sādhanā Vol. 42, No. 1, January 2017, pp. 11–21
DOI 10.1007/s12046-016-0575-7

Minimisation of total tardiness for identical parallel machine scheduling using genetic algorithm

IMRAN ALI CHAUDHRY* and ISAM A Q ELBADAWI

Department of Industrial Engineering, College of Engineering, University of Hail, Ha'il, Saudi Arabia
e-mail: i.chaudhry@uoh.edu.sa; isam149@gmail.com

MS received 17 October 2015; revised 14 July 2016; accepted 22 July 2016

Abstract. In recent years research on parallel machine scheduling has received an increased attention. This paper considers minimisation of total tardiness for scheduling of n jobs on a set of m parallel machines. A spread-sheet-based genetic algorithm (GA) approach is proposed for the problem. The proposed approach is a domain-independent general purpose approach, which has been effectively used to solve this class of problem. The performance of GA is compared with branch and bound and particle swarm optimisation approaches. Two set of problems having 20 and 25 jobs with number of parallel machines equal to 2, 4, 6, 8 and 10 are solved with the proposed approach. Each combination of number of jobs and machines consists of 125 benchmark problems; thus a total for 2250 problems are solved. The results obtained by the proposed approach are comparable with two earlier approaches. It is also demonstrated that a simple GA can be used to produce results that are comparable with problem-specific approach. The proposed approach can also be used to optimise any objective function without changing the basic GA routine.

Keywords. Parallel machine scheduling; genetic algorithm (GA); total tardiness; scheduling.

1. Introduction

Parallel machine scheduling problem is an important class of problem. It is a generalisation of the single-machine scheduling problem. The parallel machines scheduling problem can be classified as unrelated, uniform or identical parallel machines scheduling problem. In case of unrelated parallel machines, there is no relation between the processing times of the jobs and machines. This may be due to technological differences of the machines, different features of the jobs, etc. Uniform machines work at different speeds, i.e., the processing time of each job differs by a constant factor for the individual machines. However, in case of identical parallel machines scenario, all machines are identical in terms of their speed and each and every job will take the same amount of processing time on each of the parallel machines.

Parallel machine scheduling comes down to assigning each operation to one of the machines among candidate machines and then sequencing the operations assigned to a particular machine. Many real-life problems can be modelled as parallel machine scheduling problem, e.g., automotive assembly lines can assemble several non-identical or identical car models; a gas station can serve several cars simultaneously and bank counters can serve several customers at the same time.

*For correspondence

Job tardiness is a regular performance measure whereby it is always desired to finish an activity earlier, rather than later. Job tardiness is defined as the amount of completion time in excess of the due date. It is one of the most common performance measures encountered in practical problem. Minimizing total tardiness is one of the most important criteria in many manufacture systems, especially in the current situation where competition is becoming more and more intensive.

In this paper we consider scheduling of n jobs to be processed on m number of identical parallel machines to minimise total tardiness of the jobs. The rest of the paper is organised as follows: In section 2, literature review for identical parallel machine with total tardiness as objective measure is presented. Problem and assumptions are defined in section 3. Section 4 gives a brief introduction of GAs and the architecture of the proposed approach. Experimental results and discussion is presented in section 5 and finally the paper is concluded in section 6.

2. Literature review: identical parallel machine with total tardiness as objective measure

Due to its application in manufacturing, identical parallel machine total tardiness problem has received wide attention from the researchers. Researchers have used a wide variety

of algorithms, ranging from exact methods to hybrid algorithms. Root [1] presented a constructive method to minimise total tardiness in a single-stage production where all jobs have a common due date. Azizoglu and Kirca [2] proposed a branch and bound algorithm combined with an efficient lower bounding scheme to minimise total tardiness for n jobs to be processed on m identical parallel machines. The proposed algorithm can find optimal solution for problems with up to 15 jobs.

Armentano and Yamashita [3] considered scheduling of jobs on identical parallel machines to minimise mean tardiness of the jobs and present a tabu search (TS) approach for the problem. Yalaoui and Chu [4] have developed a branch-and-bound algorithm for identical parallel machines without pre-emption to minimise total tardiness. They also introduce some job dominance checks to improve the algorithm and optimally solve problem instances with up to 15 jobs and three parallel machines. Bilge *et al* [5] developed a TS algorithm to minimise total tardiness for a set of independent jobs on uniform parallel machines with sequence-dependent set-ups. The authors apply their algorithm on a set of benchmark problems and obtain considerably better results than those reported previously.

Hu [6, 7] extended the classical identical parallel machine scheduling problem by including an additional dimension of worker assignment to machines. Hu suggested that certain relationship exists between job processing time and the number of workers assigned to the job; the processing time, therefore, is no longer a constant but is related to the number of workers assigned to work on the job. Hu solves the problem in two phases. The SPES (SPT-EDD-SLACK) heuristic solves job scheduling problem, while the largest marginal contribution (LMC) procedure minimises total tardiness for worker assignment phase.

Shim and Kim [8, 9] address scheduling of n jobs on m identical parallel machines. The authors develop a branch and bound algorithm that uses dominance/pruning rules to minimise total tardiness of the jobs. Anghinolfi and Paolucci [10] proposed a hybrid meta-heuristic (HMH) approach that integrates features from TS, simulated annealing (SA) and variable neighbourhood search (VNS). Additionally, the authors also consider sequence-dependent set-ups and non-zero ready times. Shim and Kim [11] used a branch-and-bound algorithm to solve parallel machines scheduling problem. The authors consider job splitting property and assume that a job can be split into sub-jobs where the sub-jobs can be processed independently on any one of the parallel machines.

Tanaka and Araki [12] also considered identical parallel machine scheduling problem and combined branch-and-bound with Lagrangian relaxation to minimise total tardiness. The authors demonstrate that optimal solution can be obtained for up to 25 jobs and any number of machines. Biskup *et al* [13] considered the total tardiness minimisation as performance measure of a parallel machines scheduling problem, and developed a new general heuristic for finding optimal or near-optimal schedules. Chaudhry and Drake [14] also considered machine scheduling and worker assignment problems in identical parallel machines to minimise total tardiness using genetic algorithms (GAs).

Niu *et al* [15] proposed a hybrid particle swarm optimisation (PSO) for identical parallel machine scheduling with total tardiness as the objective function. The authors introduced a clonal selection algorithm (CSA) into PSO to enhance the performance. Demirel *et al* [16] investigated the parallel machine scheduling problem to minimise total tardiness and proposed a GA to solve the problem. Yalaoui [17] considered scheduling on n jobs on m identical parallel machines with job release times to minimise total tardiness and propose an exact resolution, an ant colony algorithm (ACO), a tabu search (TS) method, a set of heuristics based on priority rules and an adapted Biskup *et al* [13] (BHG) method. Wei *et al* [18] proposed a hybrid algorithm that combines variable neighbourhood descent (VND) and discrete differential evolution (DDE) algorithm to enhance local search ability. In order to accelerate the convergence of the hybrid algorithm, the initial solution for this is generated using a modified due date (MDD) constructive heuristic [19].

3. Problem formulation and assumptions

The problem considered in this paper can be formally described as scheduling n independent jobs $N = \{1, 2, \ldots, n\}$ on m identical parallel machines $M = \{1, 2, \ldots, m\}$ to minimise total tardiness. A job is tardy if its completion time C_i is greater than its due date D_i and tardiness is measured by $C_i - D_i$. If a job is not tardy, its tardiness is set to 0. Therefore, the tardiness of job i is given by max $(C_i - D_i; 0)$.

A schematic diagram for parallel machine scheduling problem with eight jobs and three machines is shown in figure 1.

The optimal solution satisfies following conditions:

(1) Each job has only one operation and a deterministic processing time.
(2) All jobs are ready at time zero.
(3) No job pre-emption (splitting is allowed);
(4) The numbers of jobs and machines are fixed.
(5) Each job has its own due date;
(6) No machine may process more than one job at a time.
(7) All machines can process all jobs.
(8) Machine set-up times are negligible.
(9) Jobs can wait for processing and each job has the same priority.
(10) Movement of jobs between machines is either negligible or included in the processing times.
(11) All machines are always available; therefore, machine breakdown is not considered.

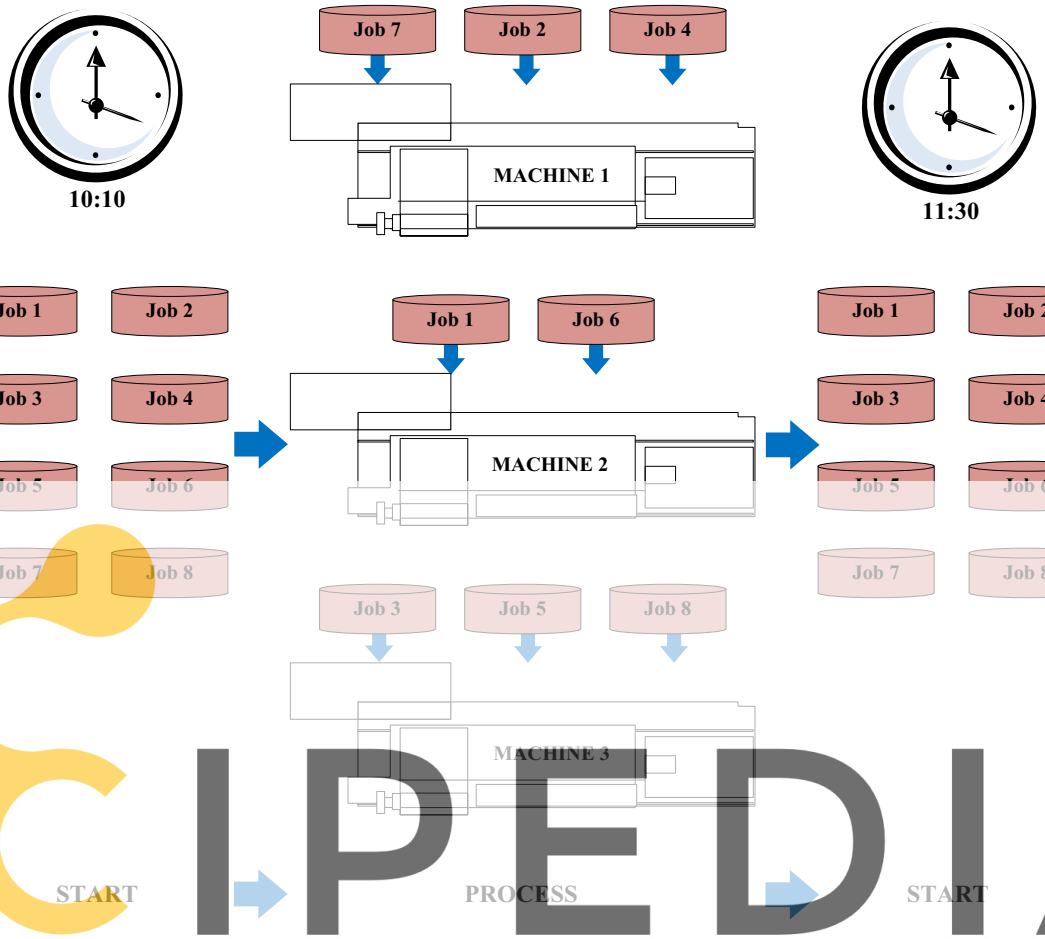

Figure 1. Schematic diagram for parallel machine scheduling.

We use the following notation to formulate the problem:

I	index of machine, $i = 1, 2, \ldots, M$; where M is the total number of parallel machines
$\pi = \{\sigma_{(1)}, \sigma_{(2)}, \sigma_{(3)}, \ldots\sigma_{(j)}, \ldots \sigma_{(k)}\}$	the sequence of jobs allocated to machine i to process
$\sigma_{(j)}$	index of job, $j = 1, 2, \ldots, k_i$, k_i represents the number of the total jobs which are processed by parallel machine i
$P_{\sigma(j)}$	the processing time of job $\sigma_{(j)}$
$S_{\sigma(j)}$	the starting time of job $\sigma_{(j)}$
$C_{\sigma(j)}$	the completion time of job $\sigma_{(j)}$
$D_{\sigma(j)}$	the due date of job $\sigma_{(j)}$
$T_{\sigma(j)}$	the tardiness of $\sigma_{(j)}$

Mathematically the parallel machine tardiness problem can be represented in the following manner:

$$S_{\sigma 1} = 0 \tag{1}$$

$$C_{\sigma(1)} = S_{\sigma(1)} + P_{\sigma(1)} \tag{2}$$

$$S_{\sigma(j)} = C_{\sigma(j-1)}; \quad j > 1 \tag{3}$$

$$C_{\sigma(j)} = S_{\sigma(j)} + P_{\sigma(j)}, \quad j > 1. \tag{4}$$

The tardiness of job $\sigma_{(j)}$ is therefore can be calculated as follows:

$$T_{\sigma(j)} = \max(C_i - D_i; 0). \tag{5}$$

The objective function in this paper is thus to minimise the total tardiness, defined as follows:

$$T = \sum_{i=1}^{M} \sum_{j=1}^{k_i} T_{\sigma(j)}. \tag{6}$$

A 0–1 mixed integer programming model for scheduling jobs on parallel machines to minimise total tardiness is given in subsequent paragraphs.

To ensure that any job could be scheduled and only once, the following constraint is formulated:

$$\sum_{i=1}^{M} \sum_{k=1}^{\text{index}} u_{(i,k,x)} = 1, \quad k = 1, 2, \ldots, N. \tag{7}$$

To ensure that each position on the list can hold only one job at the same time, the constraint is as shown below:

$$\sum_{i=1}^{M} u_{(i.k.x)} = 1, \quad k = 1, 2, \ldots, N. \tag{8}$$

The characteristics of the local variable u is therefore

$$u_{(i,k,x)} \in \{0, 1\} \quad \begin{array}{l} i = 1, 2, \ldots, M \\ k = 1, 2, \ldots, N \\ x = 1, 2, \ldots, \text{index}. \end{array} \tag{9}$$

The local variable $u_{(i,k,x)}$ can be defined as follows:

$$u_{(i,k,x)} = \left\{ \begin{array}{ll} 1 & \text{job } k \text{ is the } x\text{th position of machine } i \\ 0 & \text{otherwise} \end{array} \right\}.$$

In order to solve the identical parallel machine scheduling problem, highly sophisticated algorithms have been proposed by various researchers, whose details are given in section 2. Since the problem is NP hard, optimal solutions are often not available even for mid-size problems. Therefore, we focus on developing a GA-based solution procedure to yield efficient solutions in terms of computation time, applicability and solution quality. Equations (1)–(9) are used to develop the spreadsheet model to minimise the total tardiness for identical parallel machines.

4. Genetic algorithms

GAs are an adaptive heuristic search algorithm based on the evolutionary ideas of genetics and natural selection. GAs were first introduced by Holland [20]. GAs are the main paradigm of evolutionary computing. GAs are inspired by Darwin's theory about evolution – the "survival of the fittest". In nature, competition among individuals for scanty resources results in the fittest individuals dominating over the weaker ones.

GAs are the ways of solving problems by mimicking processes nature uses; i.e., selection, crossover, mutation and accepting, to evolve a solution to a problem. GAs are therefore adaptive heuristic search based on the evolutionary ideas of natural selection and genetics.

Since the late 1980s there has been a growing interest in GAs optimisation algorithms. GAs fall under the domain of artificial intelligence as these are based on 'evolutionary learning'. Such algorithms have been used widely for classification and learning, parameter optimisation and production scheduling. Goldberg [21] gives a detailed comprehensive introduction to GAs. Applications of GAs are found in manufacturing, engineering, phylogenetic computational science, economics, chemistry, bioinformatics, physics and numerous other fields. Detailed analysis of various application is given in Davis [22].

Schedules can improve their 'fitness' by reproduction and survival of the fittest, in exactly the same way as living creatures do. A key advantage of GAs is that they provide a 'general-purpose' schedule optimiser, with the peculiarities of any particular application being accounted for in the calculation of the fitness of a schedule without disturbing the logic of the standard GA. This means that it is a relatively straightforward matter to adapt the software implementation of the method to meet the needs of a particular application.

Davis [23] presented one of the earliest application of GA in scheduling. A recent review of GAs application in production scheduling has been presented by Chaudhry and Drake [14, 24], Chaudhry [25] and Nanvala [26].

Evolver [27], a commercial GA has been used in this paper. Evolver functions as an add-in Microsoft Excel™ spreadsheet. The shop models for identical parallel machines are built using spreadsheet's built-in functions using Eqs. (1)–(9). The evolver generates a population of trial solutions corresponding to the population size. The solution is continually improved by the GA in each successive generation. In each generation, every possible solution becomes an organism with 'genes' that can 'breed' with other organisms within the population. The spreadsheet model for identical parallel machines acts as an environment for the organisms. Fitness of survival for each organism is based on its result. The adjustable cells or variables, objective function and constraints are specified within the spreadsheet environment. Figure 2 shows the spreadsheet-evolver construction.

Various benefits have been accrued from the proposed approach. The program runs in the background, thus freeing the user to work in the foreground. Furthermore, the familiar spreadsheet layout helps the user to use the software easily and also easily carry out what-if analysis. Evolver™ has been used for various problems such as scheduling [14, 24, 25, 28–37], maintenance [38, 39], resource optimisation [40], sampling control charts [41], optimal integration of test orders in object-oriented systems [42], decision support systems [43, 44] and site precast layout arrangement [45].

4.1 *GA components*

4.1a *Chromosome representation*: Single-stage identical parallel machine scheduling problem has been addressed in

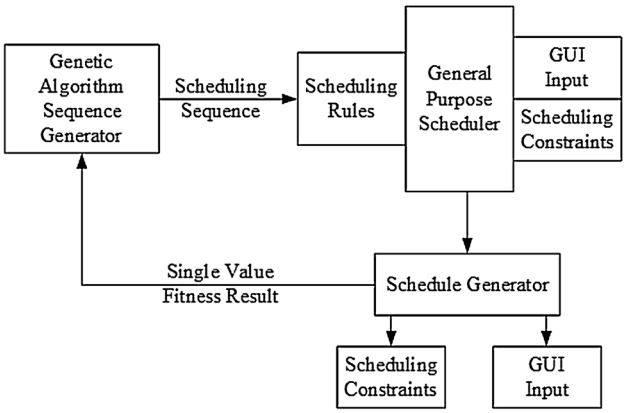

Figure 2. Spreadsheet-evolver construction.

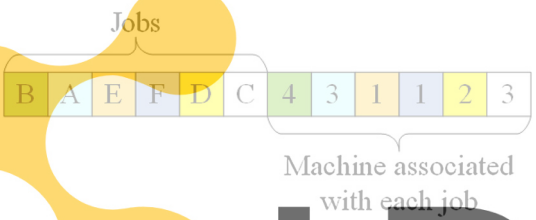

Figure 3. Chromosome representation for single-stage identical parallel machine model.

this paper. The chromosome for the problem consists of two parts: part 1 is the list of jobs while part 2 has the machine associated to each job. An example of six jobs A–F to be processed on four identical parallel machines is given in figure 3. The chromosome contains all the jobs and machines required for each of the job.

Jobs are represented by the first six genes of the chromosome. The machine associated with each corresponding job is represented by the next six genes. The chromosome would be read as follows: machine 4 is to process job B; machine 3 to process job A; machine 1 to process job E and job F; machine 2 to process job D and finally machine 3 is to process job C. For the first six genes the objective is to find the right permutation or a sequence of jobs while for last six genes we need to find the best combination of machine for each job such that total tardiness is minimised. Each of the block is calculated at a different location within the spreadsheet and linked together to calculate the objective function, i.e., total tardiness.

4.1b *Reproduction/selection*: Steady-state reproduction as proposed by GENITOR GA [46] has been implemented in Evolver GA. Unlike generational replacement technique where all parents in the population are replaced by child solutions, steady-state reproduction replaces only one organism in each iteration. As compared to generational replacement where many genes are lost as many of the best individuals may not produce at all, the steady-state

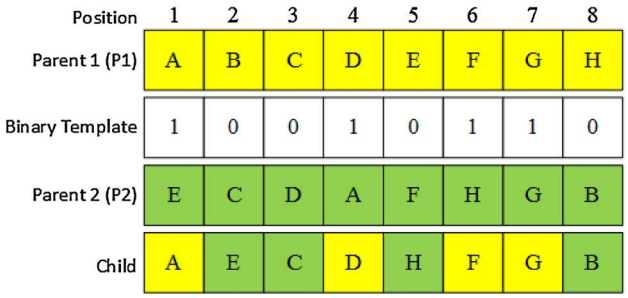

Figure 4. Order crossover.

reproduction ensures that all genes are not lost. This is a better model of what happens in longer lived species. This not only gives rise to competition among population members but also allows parents to nurture and teach their offspring [46].

Objective function value: total tardiness in this case is used to measure the fitness of a particular chromosome. Evolver GA uses a 2rank-based mechanism for parent selection. The selection is based on roulette wheel selection, whereby each parent gets a slot that is proportional to its fitness. This avoids premature convergence of GA and prevents good organisms from completely dominating the evolution from an early point.

4.1c *Crossover operator*: Crossover is the selection of two parents to produce a child solution. It is an important element of GAs. For the first six genes in figure 3, i.e., jobs order crossover developed by Davis [22] is used. The idea behind this operator is to preserve the relative order of the genes.

Figure 4 illustrates an order crossover. Order crossover is based on generating a 0–1 bit string template to determine the contribution of each parent towards the child solution. Wherever the binary template contains "1", the elements are copied from parent *P*1 into the child solution in the same position as they appear in parent *P*1. The elements associated with "0" in the binary template appear in the same order in the child solution as they appear in parent *P*2.

In figure 4, elements A, D, F and G are associated with "1" in parent *P*1. The child solution inherits these elements in the same positions as they appear in parent *P*1. Elements B, C, E and H are associated with bit string "0" and they appear in the same order in the child solution as they appear in parent *P*2.

For genes 7–12 in figure 3, i.e., machine assignment, uniform crossover is implemented. Uniform crossover works better than one- and two-point crossovers [22]. Figure 5 describes uniform crossover. A fixed mixing ratio between two parents is used in this type of crossover, thus enabling each parent chromosomes to contribute the gene level rather than at the segment level.

Consider two parents in figure 5. Parent *P*1 has been coloured yellow while parent *P*2 green. A random mask is

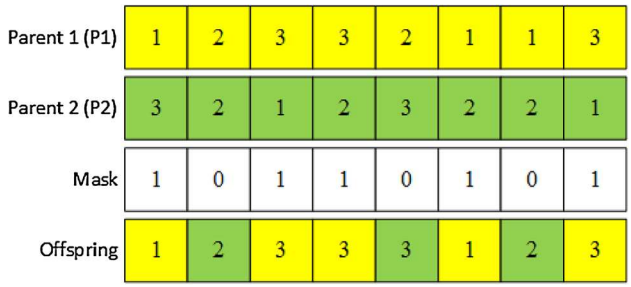

Figure 5. Uniform crossover.

generated corresponding to the crossover rate. For a crossover rate of 0.6, approximately five genes in the child solution will be contributed by parent $P1$ while three genes would be contributed by parent $P2$. A random 0–1 bit string is generated corresponding to the crossover rate. The genes corresponding to bit "1" are taken from parent $P1$ while the genes corresponding to bit "0" are taken from parent $P1$. In figure 5, the colour of the child chromosome represents the mixing of genes if the crossover rate is 0.6.

For genes 7–12 in figure 3, i.e., machine assignment, there is a set of variables that is to be adjusted and varied independent of each other. For this type of situation, a uniform crossover is preferred as it preserves schema and can generate any schema from the two parents.

4.1d *Mutation operator*: The mutation operator ensures diversity in the population. Mutation prevents the GA becoming trapped at "local optima" or "blind corners" by giving random movement about the search space. GA in this research performs order-based mutation for genes 1–6, i.e., job ordering in figure 3. In this mutation, two genes or jobs are randomly selected and their positions swapped. The "mutation rate" determines the probability that mutation is applied after a crossover. The number of swaps is decreased or increased proportionately to the decrease or increase in the mutation rate setting.

For genes 7–12 in figure 3, the mutation is performed by looking at each variable individually. This is done by randomly generating a number between 0 and 1 for each of the genes. If a gene gets a number that is less than or equal to the mutation rate (for example, 0.06), then the gene is mutated. Mutating a variable involves replacing it with a randomly generated value (within its valid min–max range).

5. Experimental results

The performance of proposed GA approach is evaluated against a set of benchmark problems proposed by Tanaka and Araki [12]. The benchmark problems are available online at http://turbine.kuee.kyoto-u.ac.jp/~tanaka/index-e.html. Tanaka and Arkai have used the standard method by Fisher [47] to generate these benchmark problems. First, the integer processing times $p_j(1 \le j \le n)$ are generated by the uniform distributions in [1, 100]. Then, let $P = \sum_{j=1}^{n} p_j$ and the integer due dates $d_j(1 \le j \le n)$ are generated by the uniform distributions in $[P(1 - \tau - R/2)/m, P(1 - \tau + R/2)/m]$. The number of jobs n and the number of machines m are changed as follows: $n = 20, 25$; $m = 2, 3, 4, 5, 6, 7, 8, 9, 10$; the tardiness factor τ and the range of due dates $R$are changed as follows: $\tau = 0.2, 0.4, 0.6, 0.8, 1.0$; and $R = 0.2, 0.4, 0.6, 0.8, 1.0$. For every combination of n, m, τ and R, five problem instances are generated. Thus, for each combination of n and m, 125 problem instances are generated. Hence a total of 2250 problems are solved. Optimal solutions for the data set are given by Tanaka and Araki [12] and available on the web site.

The problems have been simulated on a 2.7 GHz Core 2 Duo computer with 2 GB RAM. The results are based on 20 runs for each problem. For each of the run, the crossover and mutation rate of 0.65 and 0.005, respectively, are used. The population size used is 65. The GA parameters have been selected based upon earlier findings by Chaudhry and Drake [24], where an empirical analysis to study the effect of crossover rate, mutation rate and the population size was carried out. The study demonstrates that GA performance is insensitive to the crossover and mutation rate. Rather than a more precise value, the mutation rate possessed a "range" over which its value is suitable. Furthermore, provided the mutation rate is in the "good" range the performance of GA is fairly insensitive to population size. Stopping criterion for the proposed algorithm is 100,000 trials that correspond to 2 min on a Core 2 Duo 2.7 GHz computer having 2 GB RAM.

The performance of proposed GA approach is compared with branch-and-bound algorithm with Lagrangian relaxation proposed by Tanaka and Araki [12] and a hybrid PSO algorithm combining clonal selection and PSO (CSPSO) proposed by Niu *et al* [15]. Niu *et al* compared the performance of CSPSO with branch-and-bound by Tanaka and Araki for 2nd and 4th instance only and for $m = 2, 4, 6, 8$ and 10 and $n = 20$. Thus a total of 250 problems are solved by Niu *et al* The comparative performance of CSPSO and proposed GA approach against optimal solutions are given in table 1.

From table 1 we can see that CSPSO found the optimal solution for 237 problems (94.80% optimal solutions) as compared with the proposed GA, which found the optimal solution for 234 problems (93.60% optimal solutions). We can deduce from the table that the performance of the proposed approach is comparable with that of CSPSO. GA approach proposed in this study found worse solutions for 10 problems as compared with CSPSO while for eight problems the proposed approach found better solutions than did CSPSO.

Niu *et al* [15] solved only a small portion of benchmark problems given by Tanaka and Araki [12]. All 1125

Table 1. Comparison of proposed GA approach with CSPSO for $n = 20$.

	Comparison of proposed GA with CSPSO				Number of optimal solutions	
No of machines	Total problems	Same	Worse	Better	Niu *et al*	This GA
$m = 2$	50	46	2	2	48	48
$m = 4$	50	44	3	3	44	44
$m = 6$	50	48	1	1	49	48
$m = 8$	50	46	4	–	49	45
$m = 10$	50	48	–	2	47	49
Total	250	232	10	8	237	234

Table 2. Comparison of proposed GA approach with optimal solutions [12] for $n = 20$.

No of machines	Total problems	Total optimal solution found	Optimal solutions (%)	Avg error (%)	Time to reach the best solution (s)		
					Average	Minimum	Maximum
$m = 2$	125	121	96.80	0.8132	32	1	152
$m = 3$	125	121	96.80	0.0185	39	1	153
$m = 4$	125	113	90.40	0.8137	38	1	152
$m = 5$	125	112	89.60	0.5196	39	1	157
$m = 6$	125	116	92.80	2.8320	34	1	123
$m = 7$	125	116	92.80	3.8294	34	1	152
$m = 8$	125	119	95.20	0.6271	30	1	150
$m = 9$	125	120	96.00	0.7519	19	1	158
$m = 10$	125	120	96.00	0.7407	21	1	154
Total	1125	1058	94.04	1.2162	32	1	150

benchmark problems given by Tanaka and Araki for $n = 20$ were attempted with the proposed GA. Table 2 gives comparative results of the two approaches. The average, minimum and maximum times to reach the best solutions are also given in table 2.

From table 2 we can see that the proposed GA was able to find optimal solution for 1058 problems (94.04%). The average, minimum and maximum times to find the best solutions are 32, 1 and 150 s, respectively.

Tanaka and Araki [12] also give benchmark problems for $n = 25$. The same were also attempted with the proposed GA approach. Table 3 gives comparative results of the proposed approach and the optimal solutions. The proposed GA was able to find optimal solutions for 840 problems (74.67% problems) out of a total of 1125 benchmark problems.

From table 3 we can see that the percentage of optimal solutions found was better for higher number of parallel machines as compared to lower number of parallel machines. The average, minimum and maximum times to find the best solutions are 51, 2 and 156 s, respectively.

Keeping in view the general purpose nature of the proposed approach and the computational results, it can be concluded that the proposed approach generates high-quality schedules in a timely fashion. Also the qualities of solutions found by the proposed approach are comparable to those from the previous approach for the same problems.

6. Evaluation of proposed GA approach for other performance metrics

In order to check the robustness of the proposed GA approach, we present evaluation of the proposed GA approach for other performance metrics. In this regard, we address minimisation of makespan for the no-wait flowshop scheduling problem.

The performance of the proposed GA approach was compared to the performances of six other algorithms as follows.

A1: Heuristic algorithm based on heuristic preference relations and job insertion proposed by Rajendran [48]

A2: Variable neighbourhood search (VNS) procedure proposed by Schuster and Framinan [49]

A3: Hybrid algorithm based on GA and simulated annealing (GASA) proposed by Schuster and Framinan [49]

A4: TS algorithm proposed by Grabowski and Pempera [50]

Table 3. Comparison of proposed GA approach with optimal solutions [12] for $n = 25$.

No of machines	Total problems	Total optimal solution found	Optimal solutions (%)	Avg error (%)	Time to reach the best solution (s)		
					Average	Minimum	Maximum
$m = 2$	125	108	86.40	0.2765	40	1	152
$m = 3$	125	90	72.00	0.5013	47	1	158
$m = 4$	125	81	64.80	0.6424	50	1	158
$m = 5$	125	72	57.60	1.0993	50	1	133
$m = 6$	125	85	68.00	1.8320	52	1	161
$m = 7$	125	94	75.20	1.2100	67	4	160
$m = 8$	125	96	76.80	1.4552	51	3	161
$m = 9$	125	99	79.20	1.1942	60	3	159
$m = 10$	125	115	92.00	1.0224	43	3	161
Total	1125	840	74.67	1.0259	51	2	156

Table 4. Performance comparison of different meta-heuristics.

Instance	$n \times m$	Opt/ best known	A1	A2		A3		A4		A5		A6		Proposed GA	
				Min	Diff (%)	Min	Diff (%)	Min	Diff (%)	Min	Diff (%)	Min	Diff (%)	Min	Diff (%)
Problem 1	12 × 6	6921	–	–	–	–	–	–	–	–	–	–	–	6850	−1.03
Problem 2	7 × 5	683	–	–	–	–	–	–	–	–	–	–	–	565	−17.28
Problem 3	6 × 5	565	–	–	–	–	–	–	–	–	–	–	–	471	−16.64
car1	11 × 5	8142	8288	8201	0.72	8142	0	8142	0	8142	0	8142	0	8142	0
car2	13 × 4	8242	8610	8256	0.17	8242	0	8242	0	8242	0	8242	0	8242	0
car3	12 × 5	8866	9226	8866	0.00	8866	0	8866	0	8866	0	8866	0	8866	0
car4	14 × 4	9195	10,119	9548	1.86	9195	0	9195	0	9195	0	9195	0	9195	0
car5	10 × 6	9159	10,039	9496	3.68	9159	0	9159	0	9159	0	9159	0	9159	0
car6	8 × 9	9690	10,161	9690	0	9690	0	9690	0	9690	0	9690	0	9690	0
car7	7 × 7	7705	7903	7705	0	7705	0	7705	0	7705	0	7705	0	7705	0
car8	8 × 8	9372	9515	9372	0	9372	0	9372	0	9372	0	9372	0	9372	0
rec01	20 × 5	–	1590	1546	−2.77	1527	−3.96	1526	−4.03	1530	−3.77	1526	−4.03	1532	−3.65
rec03	20 × 5	–	1457	1394	−4.32	1392	−4.46	1361	−6.59	1361	−6.59	1361	−6.59	1364	−6.38
rec05	20 × 5	–	1637	1522	−7.03	1524	−6.90	1511	−7.70	1516	−7.39	1511	−7.70	1520	−7.15
rec07	20 × 10	–	2119	2070	−2.31	2046	−3.45	2042	−3.63	2042	−3.63	2042	−3.63	2088	−1.46
rec09	20 × 10	–	2141	2090	−2.38	2045	−4.48	2042	−4.62	2043	−4.58	2042	−4.62	2058	−3.88
rec11	20 × 10	–	1946	1916	−1.54	1881	−3.34	1881	−3.34	1881	−3.34	1881	−3.34	1906	−2.06

A5: Accelerated job insertion and swap method proposed by Li *et al* [51]

A6: Hybrid GA proposed by Tseng and Lin [52].

A summary of the results from the makespan criterion is given in table 4. First three problems, i.e., Problem 1, Problem 2 and Problem 3 have been taken from Zhu *et al* [53], Li and Wu [54] and Li *et al* [51], respectively. Eight problems with varying job and machine sizes (car1–car8) have been taken from Carlier [55] while six problems having 20 jobs (rec01, rec03, rec05, rec07, rec09 and rec11) have been taken from Reeves [56]. In table 4, "Instance" represents the problem name, "*n x m*" denotes the "number of jobs and number of machines, respectively, "Opt / Best Known" gives the makespan of the optimal solution of Carlier's benchmark problems.

The results found by the proposed approach for each of the instance are given in the last column. The best makespan value found by each algorithm is denoted by "Min". "% diff" represents the percentage relative difference $(C_{min} - C^*)/C^* \times 100\%$, where C^* is the makespan of the known optimal solution given by Carlier's [55]; however C^* is the makespan of the best solution found by RAJ

Table 5. Performance comparison of different meta-heuristics for larger size problems.

			A2		A3		A4		A5		A6		Proposed GA	
Instance	$n \times m$	A1	Min	Diff (%)	Min	Diff (%)	Min	Diff (%)	Min	Diff (%)	Min	Diff (%)	Min	Diff (%)
rec13	20 × 15	2709	2553	−5.76	2556	−5.65	2545	−6.05	2545	−6.05	2545	−6.05	2565	−5.32
rec15	20 × 15	2691	2532	−5.91	2529	−6.02	2529	−6.02	2529	−6.02	2529	−6.02	2565	−4.68
rec17	20 × 15	2740	2599	−5.15	2590	−5.47	2587	−5.58	2587	−5.58	2587	−5.58	2599	−5.15
rec19	30 × 10	3157	2918	−7.57	2895	−8.30	2850	−9.72	2868	−9.15	2850	−9.72	2952	−6.49
rec21	30 × 10	3015	2888	−4.21	2948	−2.22	2823	−6.37	2843	−5.70	2829	−6.17	2891	−4.11
rec23	30 × 10	3030	2704	−10.76	2827	−6.70	2700	−10.89	2703	−10.80	2700	−10.89	2783	−8.15
rec25	30 × 15	3835	3626	−5.45	3732	−2.69	3593	−6.31	3616	−5.71	3593	−6.31	3662	−4.51
rec27	30 × 15	3655	3442	−5.83	3560	−2.60	3432	−6.10	3431	−6.13	3431	−6.13	3565	−2.46
rec29	30 × 15	3583	3324	−7.23	3440	−3.99	3291	−8.15	3303	−7.81	3291	−8.15	3383	−5.58
rec31	50 × 10	4631	4413	−4.71	4757	2.72	4343	−6.22	4357	−5.92	4334	−6.41	4528	−2.22
rec33	50 × 10	4770	4515	−5.35	4998	4.78	4466	−6.37	4507	−5.51	4458	−6.54	4711	−1.24
rec35	50 × 10	4718	4458	−5.51	4891	3.67	4427	−6.17	4434	−6.02	4424	−6.23	4634	−1.78
rec37	75 × 20	8979	8081	−10.00	9508	5.89	8127	−9.49	8181	−8.89	8121	−9.56	9515	8.16
rec39	75 × 20	9158	8671	−5.32	9964	8.80	8517	−7.00	8536	−6.79	8505	−7.13	9885	7.94
rec41	75 × 20	9344	8652	−7.41	9978	6.79	8520	−8.82	8602	−7.94	8505	−8.98	9888	5.82

heuristic [48] for benchmark problems given by Reeves [56].

In table 4 the results are given for small- to medium-sized problems only. Varying the problem size allows us to access algorithm scalability; therefore we ran the algorithm on bigger size problems already published in the literature. Fifteen different problems with varying job and machine numbers were solved to see the performance of the proposed approach. All instances have been taken from Reeves [56]. The performance of proposed method is compared with the same meta-heuristics as given in table 4. The results of the simulations for each of the instance are given in table 5.

From table 5 we can see that the proposed approach produces better solutions than does RAJ heuristic [48] except for the 75 × 20 problem size (instances rec37, rec39 and rec41), where the performance of the algorithm was even worse than from RAJ heuristic [48]. Although the proposed approach does not get better solutions than the five meta-heuristics, keeping in view the general purpose nature of the proposed approach, familiar spreadsheet environment and the ability to tackle any objective function without changing the base algorithm, the performance of the algorithm can be said to be robust, thus making it a general purpose scheduling approach.

7. Conclusions

In this study a general purpose genetic algorithm (GA) is proposed for scheduling in an identical parallel machine environment where the objective function is to minimise total tardiness. The GA has been implemented in a spreadsheet environment. The proposed approach can be classified as a domain-independent approach as it is customisable and easy to implement without changing the GA routine. Furthermore, the familiar spreadsheet environment also enables the user to easily carry out what-if analysis.

For $n = 20$, the proposed GA approach found optimal solution for 93.60% of the problems as compared with 94.80% for Niu *et al* [15]. Similarly, the solutions found by the proposed approach are also comparable to branch and bound approach by Tanaka and Araki [12]. The performance of proposed GA for $n = 25$ was worse as compared with Tanaka and Araki; however, being a general purpose approach, being able to be used for a wide class of problems, the results are considered to be comparable with those from previous approaches. The proposed approach was not able to find optimal solution for the problem instances presented in the preceding paragraphs. This is due to the fact that a general purpose domain-independent GA has been presented to solve a varied class of problems. The proposed approach presented here is truly general purpose as the spreadsheet model is customisable to add more jobs or machines as has been demonstrated with the benchmark problems or even change the objective function without changing the basic GA routine.

The key advantage of GAs depicted here is that they are able to provide a general purpose solution to the scheduling problem and are not problem-specific. The peculiarities of any particular scenario are accounted for in the shop model developed in the spreadsheet without disturbing the logic of the GA routine. In order to eliminate undesirable schedules, the GA can also be combined with a rule set by capturing the expertise of a human scheduler.

References

[1] Root G J 1965 Scheduling with deadlines and loss functions on k parallel machines. *Manage. Sci.* 11(3): 460–475

[2] Azizoglu M and Kirca O 1998 Tardiness minimization on parallel machines. *Int. J. Prod. Econ.* 55(2): 163–168

[3] Armentano V A and Yamashita D S 2000 Tabu search for scheduling on identical parallel machines to minimize mean tardiness. *J. Intell. Manuf.* 11(5): 453–460

[4] Yalaoui F and Chu C 2002 Parallel machine scheduling to minimize total tardiness. *Int. J. Prod. Econ.* 76(3): 265–279

[5] Bilge Ü, Kıraç F, Kurtulan M and Pekgün P 2004 A tabu search algorithm for parallel machine total tardiness problem. *Comput. Oper. Res.* 31(3): 397–414

[6] Hu P C 2004 Minimising total tardiness for the worker assignment scheduling problem in identical parallel-machine models. *Int. J. Adv. Manuf. Technol.* 23(5–6): 383–388

[7] Hu P C 2006 Further study of minimizing total tardiness for the worker assignment scheduling problem in the identical parallel-machine models. *Int. J. Adv. Manuf. Technol.* 29(1–2): 165–169

[8] Shim S O and Kim Y D 2004 Minimizing total tardiness in an identical-parallel machine scheduling problem. In: *Proceedings of the fifth Asia Pacific industrial engineering and management systems conference*, Gold Coast, Australia

[9] Shim S O and Kim Y D 2007 Scheduling on parallel identical machines to minimize total tardiness. *Eur. J. Oper. Res.* 177(1): 135–146

[10] Anghinolfi D and Paolucci M 2007 Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach. *Comput. Oper. Res.* 34(11): 3471–3490

[11] Shim S O and Kim Y D 2008 A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property. *Comput. Oper. Res.* 35(3): 863–875

[12] Tanaka S and Araki M 2008 A branch-and-bound algorithm with Lagrangian relaxation to minimize total tardiness on identical parallel machines. *Int. J. Prod. Econ.* 113(1): 446–458

[13] Biskup D, Herrmann J and Gupta J N D 2008 Scheduling identical parallel machines to minimize total tardiness. *Int. J. Prod. Econ.* 115(1): 134–142

[14] Chaudhry I A and Drake P R 2009 Minimizing total tardiness for the machine scheduling and worker assignment problems in identical parallel machines using genetic algorithms. *Int. J. Adv. Manuf. Technol.* 42(5–6): 581–594

[15] Niu Q, Zhou T and Wang L 2010. A hybrid particle swarm optimization for parallel machine total tardiness scheduling. *Intl. J. Adv. Manuf. Technol.* 49(5–8): 723–739

[16] Demirel T, Ozkir V, Demirel N C and Tasdelen B 2011 A genetic algorithm approach for minimizing total tardiness in parallel machine scheduling problems. In: *Proceedings of the World Congress on Engineering*, 2011, London, UK

[17] Yalaoui F 2012 Minimizing total tardiness in parallel-machine scheduling with release dates. *Int. J. Appl. Evol. Comput.* 3(1): 21–46

[18] Wei M, Deng G L, Xu Z H and Gu X S 2012 Parallel machine tardiness scheduling based on improved discrete differential evolution. *Adv. Mat. Res.* 459: 266–270

[19] Baker K R and Bertrand J W M 1982 A dynamic priority rule for scheduling against due-dates. *J. Oper. Manage.* 3(1): 37–42

[20] Holland J H 1975 *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI

[21] Goldberg D E 1989 *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA

[22] Davis L 1991 *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold

[23] Davis L 1985 Job shop scheduling with genetic algorithms. In: *Proceedings of the 1st international conference on genetic algorithms*, L. Erlbaum Associates Inc

[24] Chaudhry I A and Drake P R 2008 Minimizing flow-time variance in a single-machine system using genetic algorithms. *Int. J. Adv. Manuf. Technol.* 39(3–4): 355–366

[25] Chaudhry I A 2010 Minimizing flow time for the worker assignment problem in identical parallel machine models using GA. *Int. J. Adv. Manuf. Technol.* 48(5–8): 747–760

[26] Nanvala H 2011 Use of genetic algorithm based approaches in scheduling of FMS: a review. *Int. J. Eng. Sci. Technol.* 3(3): 1936–1942

[27] Evolver: the genetic algorithm super solver, V 1998 New York, USA: Palisade Corporation

[28] Chaudhry I A 2012a A genetic algorithm approach for process planning and scheduling in job shop environment. In: *Proceedings of the World Congress on Engineering*, 2012, London, UK

[29] Chaudhry I A 2012b Job shop scheduling problem with alternative machines using genetic algorithms. *J. Central South Univ.* 19(5): 1322–1333

[30] Chaudhry I A and Mahmood S 2012 No-wait flowshop scheduling using genetic algorithm. In: *Proceedings of the World Congress on Engineering*, 2012, London, UK

[31] Hayat N and Wirth A 1997 Genetic algorithms and machine scheduling with class setups. *Int. J. Comput. Eng. Manage.* 5(2): 10–23

[32] Hegazy T and Elbeltagi E 2000 Simplified spreadsheet solutions II: overall schedule optimization. *J. Construct. Eng. Manage.* 127(6): 469–475

[33] Jeong S J, Lim S J and Kim K S 2006 Hybrid approach to production scheduling using genetic algorithm and simulation. *Int. J. Adv. Manuf. Technol.* 28(1–2): 129–136

[34] Nassar K 2005 Evolutionary optimization of resource allocation in repetitive construction schedules. *ITcon* 10: 265–273

[35] Ruiz R and Maroto C 2001 Flexible manufacturing in the ceramic tile industry. In: *Proceedings of the eighth international workshop on project management and scheduling*, Valencia, Spain

[36] Sadegheih A 2007 Sequence optimization and design of allocation using GA and SA. *Appl. Math. Comput.* 186(2): 1723–1730

[37] Shiue Y R and Guh R S 2006 Learning-based multi-pass adaptive scheduling for a dynamic manufacturing cell environment. *Robot. Comput. Integr. Manuf.* 22(3): 203–216

[38] Saranga H and Kumar U D 2006 Optimization of aircraft maintenance/support infrastructure using genetic algorithms – level of repair analysis. *Ann. Oper. Res.* 143(1): 91–106

[39] Shum Y S and Gong D C 2007 The application of genetic algorithm in the development of preventive maintenance analytic model. *Int. J. Adv. Manuf. Technol.* 32(1–2): 169–183

[40] Hegazy T and Kassab M 2003 Resource optimization using combined simulation and genetic algorithms. *J. Construct. Eng. Manage.* 129(6): 698–705

[41] He D and Grigoryan A 2002 Construction of double sampling s-control charts for agile manufacturing. *Qual. Reliab. Eng. Int.* 18(4): 343–355

[42] Briand L C, Feng J and Labiche Y 2002 Using genetic algorithms and coupling measures to devise optimal integration test orders. In: *Proceedings of the 14th international conference on software engineering and knowledge engineering*, ACM, Ischia, Italy, pp. 43–50

[43] Barkhi R, Rolland E, Butler J and Fan W 2005 Decision support system induced guidance for model formulation and solution. *Decis. Support Syst.* 40(2): 269–281

[44] Eusuff M, Ostfeld A and Lansey K 2000 An overview of HANDSS: Hula aggregated numerical decision support system. In: *Proceedings of building partnerships*, American Society of Civil Engineers, pp. 1–6

[45] Cheung S O, Tong T K L and Tam C M 2002 Site pre-cast yard layout arrangement through genetic algorithms. *Automat. Construct.* 11(1): 35–46

[46] Whitley D and Kauth K 1988 GENITOR: a different genetic algorithm. In: *Proceedings of the 1988 Rocky Mountain conference on artificial intelligence*

[47] Fisher M 1976 A dual algorithm for the one-machine scheduling problem. *Math. Program.* 11(1): 229–251

[48] Rajendran C 1994 A no-wait flowshop scheduling heuristic to minimize makespan. *J. Oper. Res. Soc.* 45(4): 472–478

[49] Schuster C J and Framinan J M 2003 Approximative procedures for no-wait job shop scheduling. *Oper. Res. Lett.* 31(4): 308–318

[50] Grabowski J and Pempera J 2005 Some local search algorithms for no-wait flow-shop problem with makespan criterion. *Comput. Oper. Res.* 32(8): 2197–2212

[51] Li X, Wang Q and Wu C 2008 Heuristic for no-wait flow shops with makespan minimization. *Int. J. Prod. Res.* 46(9): 2519–2530

[52] Tseng L Y and Lin Y T 2010 A hybrid genetic algorithm for no-wait flowshop scheduling problem. *Int. J. Prod. Econ.* 128(1): 144–152

[53] Zhu X, Li X and Wang Q 2008 Hybrid heuristic for *m*-machine no-wait flowshops to minimize total completion time. In: Shen W, Yong J, Yang Y, Barthès J P and Luo J (Eds) *Computer supported cooperative work in design IV*, vol 5236. Springer, Berlin–Heidelberg, pp. 192–203

[54] Li X and Wu C 2008 Heuristic for no-wait flow shops with makespan minimization based on total idle-time increments. *Sci. China Ser. F: Inf. Sci.* 51(7): 896–909

[55] Carlier J 1978 Ordonnancements a contraintes disjonctives. *R.A.I.R.O. Recherche operationelle/Oper. Res.* 12(4): 333–350

[56] Reeves C R 1995 A genetic algorithm for flowshop sequencing. *Comput. Oper. Res.* 22(1): 5–13